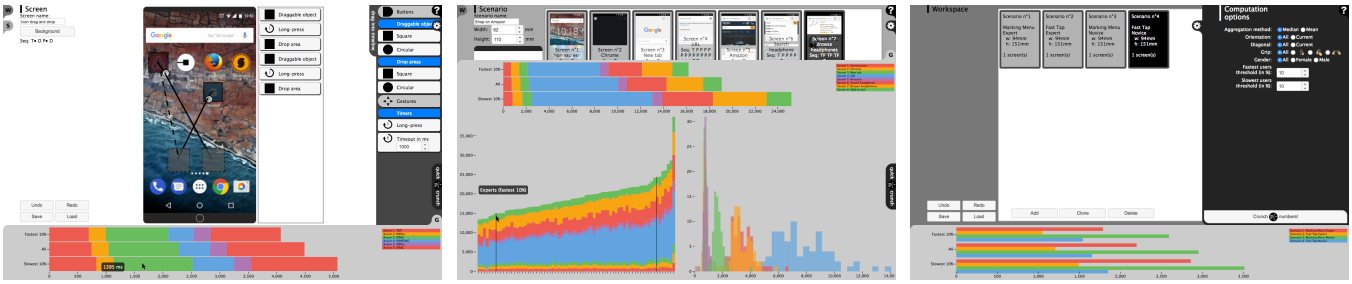
Storyboard-Based Empirical Modelling of Touch Interface Performance

Alix Goguey, Géry Casiez, Andy Cockburn, Carl Gutwin

Storyboard-Based Empirical Modelling of Touch Interface Performance

Alix Goguey[1], **Géry Casiez**[2,3], **Andy Cockburn**[4] **& Carl Gutwin**[1]
[1]University of Saskatchewan, Canada, [2]Université de Lille, France, [3]Inria, France,
[4]University of Canterbury, New Zealand
alix.goguey@usask.ca, gery.casiez@univ-lille.fr, andy@cosc.canterbury.ac.nz, gutwin@cs.usask.ca



(a) Example of interaction sequence (b) Example of predictions (c) Example of scenario comparison

Figure 1: Illustration of StEM: (a) users drag and drop actions onto a timeline to construct an interaction sequence; (b) users can visualize prediction times for a scenario composed of different screens; (c) users can compare scenarios, and filter the predictions according to factors such as screen size.

ABSTRACT

Touch interactions are now ubiquitous, but few tools are available to help designers quickly prototype touch interfaces and predict their performance. For rapid prototyping, most applications only support visual design. For predictive modelling, tools such as CogTool generate performance predictions but do not represent touch actions natively and do not allow exploration of different usage contexts. To combine the benefits of rapid visual design tools with underlying predictive models, we developed the Storyboard Empirical Modelling tool (StEM) for exploring and predicting user performance with touch interfaces. StEM provides performance models for mainstream touch actions, based on a large corpus of realistic data. We evaluated StEM in an experiment and compared its predictions to empirical times for several scenarios. The study showed that our predictions are accurate (within 7% of empirical values on average), and that StEM correctly predicted differences between alternative designs. Our tool provides new capabilities for exploring and predicting touch performance, even in the early stages of design.

Author Keywords

Touch interaction; Modelling; Performance prediction.

ACM Classification Keywords

H.5.2 Information interfaces (e.g. HCI): User interfaces

INTRODUCTION

Touch interfaces are now a ubiquitous means for interaction with computers. However, despite this ubiquity, it can be difficult for designers and researchers to predict how different designs for touch-based interactions will perform, and there are few tools available for exploring the performance of touch prototypes. For example, a designer may wonder whether interaction time for a common task will be reduced by adding a "bezel swipe" menu, compared to a regular menu design – and may wonder whether any performance improvement will be consistent across different screen sizes, screen orientations, and different ways of holding the device.

Several tools exist for supporting the visual design of touch interfaces, but these systems do not include performance prediction. For predictive modelling, applications such as CogTool [15] can generate predictions based on frameworks including GOMS/KLM – but these systems do not incorporate touch actions natively, and instead approximate touch using existing mouse-based models. Recent work has provided some touch-based extensions to KLM-style models: for example, with new operators for touch actions like tapping, pointing, dragging, and flicking [29], and with time estimates for a few basic "fingerstroke-level" actions [18]. Even with these extensions, however, modelling of touch-based interactions is still limited: there is no modelling framework that provides time estimates for the full set of touch operators; in addition, the most complete existing framework was designed to predict actions in games, covers only a single device and orientation, and only provides estimates for users who are working as fast as they can [18].

As a result, it is still difficult for researchers and designers to predict user performance with touch UIs, and very difficult to do so for early design prototypes. To combine the benefits of rapid visual design tools with the capability of predictive models, we developed a new interactive tool – the Storyboard Empirical Modelling tool (StEM) – for exploring and predicting user performance with touch interfaces. StEM provides predictive models of six mainstream touch actions (tap, point, drag, swipe, scale, and rotate), based on a large cor-

pus of realistic data gathered from several device types, hand grips, and screen orientations. The corpus was built from a field experiment that has gathered 39,804 touch tasks from 201 users to date, and the corpus continues to grow. We characterized and compared different factors within this dataset, and found that there are significant performance differences between different devices, different screen orientations, and different hand grips, showing that designers can benefit from predictions that cover a wider range of usage situations.

We evaluated StEM's corpus-based predictive capabilities using a lab study where participants carried out basic tasks under controlled conditions, and also completed full scenarios that we had previously modelled using StEM (including an e-commerce selection task, a map manipulation task, an e-mail task, and a navigation task with a multi-level settings menu). We found that our predictions closely matched the empirical results, in terms of interface comparisons, crossover points, and actual performance values. Our absolute time predictions were on average within 7% of empirical results (1.1s out of a 15-seconds task), and never worse than 13%; this was more accurate than the results of our best-effort approximation using either CogTool or FLM. We also replicated an empirical comparison between Marking Menus and FastTap menus from prior research; our predictions were again very close to the published values, and correctly identified the performance order of the different techniques.

The StEM tool provides new power for designers and researchers who need to understand user performance with touch interfaces. At any stage in the development of touch prototypes, StEM provides comprehensive and accurate estimates of touch performance, and it provides these estimates for a wide range of usage situations involving different types of users, devices, and postures. In addition, our corpus of empirical data and our design tool are publicly available[1], and can be used by anyone who needs to more thoroughly explore touch performance.

RELATED WORK

Extending the Keystroke Level Model

The Keystroke-Level Model (KLM) [4] was developed to predict the time performance of interaction on desktop computers using a mouse and a keyboard. It describes an interaction as a series of atomic actions, each represented by an operator. For instance, the action of deleting an icon would be described as follows: mentally prepare for the task (M, which represents the user's thinking or decision-making process), find the icon (M), point at the icon (P), press and release the mouse button (BB), move the hand to the keyboard (H) and press the delete key (K). A total of 2 M operators, 1 P, 2 B, 1 H and 1 K combine to yield a prediction time of 4.6 s.

However, with the rise of touch-based interfaces, KLM required extension to represent an additional set of atomic actions. Holleis *et al.* [13] extended the KLM operators for button-based mobile phones (without touchscreens), and Li *et al.* added operators to cover stylus interaction on mobile devices [19]. The KLM model was then extended to touchscreen interaction [18, 29].

[1] ns.inria.fr/mjolnir/StEM/

First, Rice *et al.* introduced operators specific to touch interaction, such as tap, drag, pinch, zoom, rotate, gesture, swipe, and tilt [29]. These operations, along with a subset of KLM operators, formed their Touch-Level Model (TLM). Their work only described the model, rather than providing timing information for the actions. Second, time estimates for some of the new operators were introduced by Lee *et al.* [18] as part of their Fingerstroke-Level Model (FLM). Lee *et al.* carried out an experiment to empirically determine the time of four of the new operators: tapping, pointing, dragging, and flicking. However, the focus of their work was on helping mobile game designers improve game mechanisms, so the study was conducted using only a single device and orientation: a 4.3-inch screen in landscape mode, where participants used their non-dominant hand to hold the device and performed touch actions with their dominant thumb. In addition, participants were instructed to be as fast as possible. In contrast to this focused study, one of our main goals was to estimate touch-action times across a variety of conditions (different form factors, grips, and orientations) and in ecologically-valid usage conditions.

Modelling from low level actions to user strategies

Low level motor actions on touch interfaces have been modelled for pointing [8, 25, 30], dragging [6], as well as transformations that include rotation, scaling and translation [9, 31]. Derived models are typically based on Fitts' law [7] or variations such as FFitts law, which captures movements when accuracy matters [3]. We used these models to analyse the data of our corpus. For higher level models, researchers have also examined the strategies that users follow to perform a task, including consideration of how interaction techniques can affect these strategies [2, 11, 21].

Modelling touch interaction

Giving early insights about interface performance should help designers to identify strengths and weaknesses of different alternatives, and consequently improve design. Existing wire-framing tools such as Balsamiq©, AxureRP©, InVision©, SILK [17], Marquise [26] and FrameWire [20] allow rapid prototyping and storyboarding of UI/UX design elements and sequences, and already enable exploration of an application's design space without having to carry out full implementation. However, wire-frame designs do not provide much information about performance: they require an experienced practitioner to recognize flaws in the design, and because they don't provide time estimates, it can be difficult to make reasoned choices about competing alternatives.

The use of predictive models like KLM or FLM assist the iterative design process by providing performance estimates - helping designers identify which alternatives are better suited to a given usage context. For example, Quaresma *et al.* used KLM to compare iPhone navigation applications and assess the visual demand on drivers [28]. Models can also help designers and researchers compare interaction techniques and obtain early insights on performance: for example, Goguey *et al.* used FLM to compare different interaction techniques across different scenarios on smartphones and tablets [10]. They compared a technique using finger identification and normal touch-based GUI interaction.

One main challenge in the use of predictive models is to compute the sequence of operators for a given scenario, since this process requires expertise in composing and combining different individual actions. John *et al.* aimed to automatize that modelization process by creating CogTool: "a UI prototyping tool that can produce quantitative predictions of how users will behave when the prototype is ultimately implemented" (CogTool user guide, p. 2) [15]. CogTool allows designers to create a mock-up of an application using a series of *Frame* objects, each representing a different screen. On each Frame, the user defines which part is interactive using pre-defined widget overlays. Once widgets have been specified, the user links them in order to create a sequence of interaction. Finally, CogTool computes time predictions for each sequence.

CogTool was designed to automate the use of KLM/GOMS models, but as described above, these models do not cover all aspects of touch-based interaction. The developers of CogTool have shown that it can be extended to touch interfaces, but there are limitations: for example, some actions such as swipes, rotations and pinches cannot be modeled, and others like drags need to be modeled using CogTool's existing tap-down and tap-up actions. In addition, it is not clear to what extent those touch models are directly based on the mouse behaviours that are built into the tool.

A few projects have tested CogTool's abilities in modelling touch interfaces. For example, Abdulin *et al.* [1] tested KLM's accuracy in predicting performance on medium-sized touchscreens. They used the KLM model with CogTool to predict time on 7" and 10" touchscreens, and compared to actual user time on different button-based UIs; the study showed less than 5% difference when completing pointing based interactions. Another study compared time estimates provided by CogTool to real measures on a mobile wallet application [27], and found a difference of 20% between predicted and empirical times. These results suggest that CogTool may be well-suited to predict performance of button-based UIs on touchscreens; but it is less clear whether it can capture other types of touch interaction that are farther from the underlying assumptions of mouse-and-windows designs. In addition, CogTool cannot provide time estimates across a range of users, devices, or orientations.

In the sections below we describe the StEM tool we developed to overcome these limitations in current frameworks. First, however, we describe the database of realistic touch action trials that we gathered in order to provide time estimates for our models of tapping, pointing, dragging, swiping, resizing, and rotating actions.

GATHERING TOUCH TIMES: THE TOUCH-ACTION DB

Our design tool predicts the times of touch actions based on a corpus of data about these actions. This database (the Touch-Action DB or TADB) is populated from an ongoing web-based field experiment that asks people to carry out simple touch tasks on their own devices and in everyday environments. We have gathered a large (and growing) corpus of information about the time for a wide range of touch actions, devices, orientations, and hand grips.

The field experiment uses a custom web application (figure 2) that works with any phone, tablet, or touchscreen laptop that can run a web browser. The application is hosted at a publicly-available location (ns.inria.fr/mjolnir/steam/). When participants visit this site, they are enrolled in the study (including demographics, device details, and informed consent) and then presented with a series of touch tasks. Each task involves an elementary touch-based interaction: tapping, pointing, dragging, swiping, rotating, and resizing. After a training phase where participants try out each of the interactions, users are given test tasks in random order and asked to carry out the tasks as they would in everyday use. Participants are allowed to stop at any time (i.e., each participant completes as many tasks as wanted). We stored a web cookie so that if a participant continued the study later, they would be recognized (and they would not have to repeat the demographics and consent forms).

The tasks in the field experiment are:

- Tapping (Figure 2b). Participants are shown a circular white target at a random location (after a random 0.5s-2.5s timeout) and must tap the target with a finger. The starting location for the user's finger is uncontrolled (see discussion below). We record the time between the target's appearance and the tap.
- Pointing (Figure 2c). Participants are shown a circular start area and a circular target (positions and sizes of the circles are controlled to provide a range of IDs). Participants tap the start area and then the target area. We record the time between the first tap on the start area and the first tap on the target area.
- Dragging (Figure 2d). Participants are shown a draggable yellow circle and a circular green target; again, positions and sizes of the circles are controlled to provide a range of IDs. Participants drag the yellow circle to the green target. We record the time between the first touch on the yellow circle and its eventual release inside the green target.
- Rotation (Figure 2e). Participants are shown a yellow circle with one black mark indicating the rotational angle of the circle, and two red marks indicating a rotation target. The aperture between the red marks, and the starting angle of the circle, are controlled. Participants rotate the yellow circle, using two fingers, to place the black mark between the two red marks. We record the time between the first rotation of the yellow object and its successful release.
- Scaling (Figure 2f). Participants are shown a yellow circle and a green ring. The initial size of the yellow circle and the thickness of the green ring are controlled. Participants scale the yellow circle, using two fingers and a pinch or expand gesture, so that its edges are inside the green ring. We record the time between the first size change of the yellow object and its successful release.
- Swiping (Figure 2g). Participants are shown a draggable yellow circle at one end of a line. The position of the circle, and the direction and length of the line are controlled. Participants drag the yellow circle along the path and release it after it has moved at least 50% of the path's length or the circle's velocity is higher than a threshold (determined empirically based on typical Android behaviour). We record

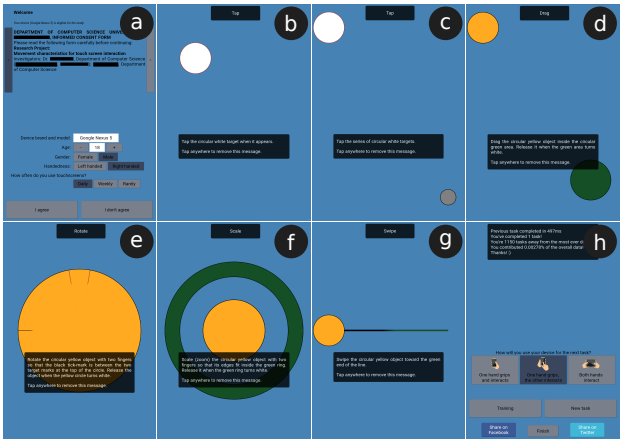


Figure 2: Web application and example tasks used to collect touch data: (a) home screen, (b) tapping task, (c) pointing task, d) dragging task, (e) rotation task, (f) scaling task, (g) swiping task, (h) task separation screen.

the time between the initial touch on the circle and its successful release. The swipe results were used for two operators in the StEM tool described below: the distance version of the motion was used for the swipe operator, and the speed version was used for the flick operator.

Between tasks, participants are presented with a screen that allows them to indicate their grip on the device (Figure 2h): $\text{GRIP}^{\text{ONE}}_{\text{HANDED}}$ (*ie.* held in the dominant hand and touched using the thumb of that hand); $\text{GRIP}^{\text{HOLD\&}}_{\text{TOUCH}}$ (*ie.* held in the non-dominant hand and touched with a finger of the dominant hand); and $\text{GRIP}^{\text{TWO}}_{\text{HANDED}}$ (held in both hands and touched with fingers/thumbs of either hand). Once the grip is selected they can complete any number of trials; time for each task is recorded in the database along with meta-data about the person, the device, the grip, and the screen orientation (portrait or landscape). The participant's grip is the only element that cannot be identified automatically by the system, and therefore is a source of potential in-correctness in the database (if participants state the wrong grip). However, we believe this problem is likely to be infrequent, since we allowed participants to indicate a grip change before any trial; we also attempted to limit the number of unrecorded grip changes by removing tasks requiring two-finger gestures when participants had indicated they were using the one-handed grip.

The TADB website was advertised via social media, newsletters, and within classes. So far, we have gathered a total of 39,804 trials from 201 different users.

Models used for touch tasks

The empirical data gathered from the field experiment provides time estimates and regression coefficients for a set of models covering each touch action. For pointing, dragging, scaling, and rotation, we used Fitts' Law models [7, 23] that were developed in previous work [9]. Our models use the Shannon formulation for index of difficulty ID [22, 24]: $T = a + b\log_2(1 + \text{AMP}/\text{TOL})$ where T represents the time prediction for a given ID $\log_2(1 + \text{AMP}/\text{TOL})$ [2], AMP represents the amplitude of the task (for pointing and dragging, the distance between the circles; for rotation, the angle between the black rotation indicator and the centre of the red targets; for scaling, twice the distance between the yellow circle's edge and the middle of the green ring), and TOL represents the tolerance of the task (*ie.* for pointing, the diameter of the target; for dragging, the difference between the diameters of the target and dragged circle; for rotation, the aperture angle between the two red marks; and for scaling, the thickness of the target ring).

For tapping and swiping, no model has become established in previous literature. The tapping task requires additional consideration because the starting position of the tapping finger is unknown (it is above the screen and out of sensing range). We used the pointing model with a fixed amplitude corresponding to the diagonal of the device's screen[3]. For the swiping task, we did not use an equation at all, but rather used the simple empirical mean (or median) times from the database as a rough estimate (taking all swiping trials into account). Our models for tapping and swiping are reasonable starting points that can be easily updated as new research becomes available.

VALIDATION OF THE TOUCHDB CORPUS

In order to validate our method for gathering touch data, we conducted a controlled lab experiment. We recruited participants to perform the previously described tasks using our web application, while varying and controlling factors such as device, orientation, and hand grip.

Participants

Eighteen participants (nine men, nine women) were recruited from the local university community and were given an honorarium of $10. The average age of the participants was 24.9 (SD 5.8), and all participants used a multitouch device on a daily basis. One participant was left-handed.

Design

Participants came to the lab for the study. As for the in-the-wild version of the experiment, no instruction on speed or accuracy requirements were given in order to approach everyday behaviour, which differs from previous work [18].

The experiment varied four factors: DEVICE (5 or 7 inch screen), ORIENTATION (portrait or landscape), GRIP ($\text{GRIP}^{\text{ONE}}_{\text{HANDED}}$, $\text{GRIP}^{\text{HOLD\&}}_{\text{TOUCH}}$ or $\text{GRIP}^{\text{TWO}}_{\text{HANDED}}$), and ID (3 different TOL for tapping, 3 different TOL and 2 different AMP for pointing and dragging, 3 different TOL, and 3 different AMP for rotation and 3 different TOL and 2 different AMP for scaling). For swiping, the ID factor was replaced by the length of the path (2 different lengths). Each combination of factors was repeated several times: 8 times for tapping (randomly varying the location of the target), 8 times for pointing and dragging (varying the direction of the movement), 4 times for rotation (varying the rotation direction between clockwise and counter-clockwise), 4 times for scaling (varying the direction between contract and expand), and 4 times for swiping (varying the movement direction).

[2] We use the generic terms *Amplitude* and *Tolerance* instead of *Distance* and *Width*, as our terms cover both rotation and scaling tasks as well as pointing and dragging.

[3] We make the assumption that the hand is at rest next to the device before each tapping action and thus we use the diagonal of the screen as an estimate of the "pointing" amplitude.

Tasks	R^2	Adjusted R^2	a	b
Tapping	$> .86$	$> .84$	.53 s	.06 $s.bits^{-1}$
Pointing	$> .92$	$> .91$	.20 s	.11 $s.bits^{-1}$
Dragging	$> .87$	$> .85$	.17 s	.18 $s.bits^{-1}$
Rotation	$> .95$	$> .94$	.32 s	.30 $s.bits^{-1}$
Scaling	$> .54$	$> .45$	.74 s	.10 $s.bits^{-1}$

Table 1: Summary of the Fitts' regression analysis for all users in the database.

To keep session durations at 45 minutes, we removed three combinations of DEVICE, ORIENTATION, and GRIP that we considered as overly difficult to perform: $\text{GRIP}^{\text{ONE}}_{\text{HANDED}}$ and $\text{GRIP}^{\text{TWO}}_{\text{HANDED}}$ with a 7-inch screen, and $\text{GRIP}^{\text{ONE}}_{\text{HANDED}}$ with a 5-inch landscape-oriented screen. Once again, no rotation or scaling tasks were presented for the one-handed grip.

For each combination, each participant completed 24 trials for tapping, 48 trials for pointing and dragging, 36 trials for rotation, 24 trials for scaling, and 32 trials for swiping. Across all participants, 25,632 trials were logged; of these, 1,125 trials (4.4%) were removed as erroneous because they were performed using more than one action.

Results

Our goals in this analysis were to compare lab results (that had strong controls) to our in-the-wild TouchDB data, and also to characterize the main differences within the dataset. We examined the first issue by determining how well the data in each corpus fit the Fitts' models; we examined the second by conducting standard within-subjects RM-ANOVA tests on the measured variable TIME. We used the *ezANOVA* package in the *ez* R environment. When significant effects were found, we carried out post-hoc analyses using pairwise T-test comparisons with the Holm correction. We used median times when aggregating data, since task mean completion times are typically not normally distributed.

Task index of difficulty varies substantially across devices and screen orientation, because of the differences in available screen space. In order to be able to compare the different factors in the RM-ANOVA, we rounded the IDs to the closest integer and performed the analysis on these ID bins that were in common across all factors (1, 2, 3 and 4 for pointing, 2, 3 and 4 for dragging, 2, 3, 4 and 5 for rotation and 1, 2 and 3 for scaling).

In the following sections, we provide detailed results for pointing and dragging, as they account for the majority of the touch interactions, and summarize the analysis results for the other tasks (see tables 1 and 2 and figure 3).

Pointing

Fitts law regression – The regression analysis on the experimental data yielded an $r^2 > .98$, an intercept a of .26 s and a slope b of .09 $s.bits^{-1}$. To compare against the full TouchDB dataset, we also performed a regression analysis for all the users in the database ($r^2 > .92$, a=.20 s and b=.11 $s.bits^{-1}$) and all the users without the experimental data ($r^2 > .89$, a=.17 s and b=.12 $s.bits^{-1}$).

Main effects on TIME– As expected, we found a significant main effect of ID ($F_{3,51}$=216.7, p<.0001). Post-hoc tests showed significant differences between all levels of ID (all p<.005), with higher-ID tasks taking longer. We found a significant main effect of GRIP ($F_{2,34}$=8.1, p<.005). Post-hoc tests showed significant differences between all levels of GRIP (all p<.05): $\text{GRIP}^{\text{TWO}}_{\text{HANDED}}$ was faster than $\text{GRIP}^{\text{HOLD\&}}_{\text{TOUCH}}$, which was faster than $\text{GRIP}^{\text{ONE}}_{\text{HANDED}}$. We found no significant main effect of DEVICE ($F_{1,17}$=2.5, p=.13) or ORIENTATION ($F_{1,17}$=.5, p=.5).

Interactions on TIME– We found significant interactions between ID and each of the other factors (DEVICE: $F_{3,51}$=8.9, p<.0001; ORIENTATION: $F_{3,51}$=3.2, p<.05; GRIP: $F_{6,102}$=9.5, p<.0001). Post-hoc tests showed multiple significant differences, but none involving a specific ID across the other factors (all p>.1), except for ID 4, where the 5-inch device was faster than the 7-inch device (p<.005) and $\text{GRIP}^{\text{TWO}}_{\text{HANDED}}$ was faster than both $\text{GRIP}^{\text{ONE}}_{\text{HANDED}}$ and $\text{GRIP}^{\text{HOLD\&}}_{\text{TOUCH}}$ (all p<.01). These results suggest that as a task gets harder, DEVICE and GRIP matter more. We found no significant interaction between ORIENTATION and GRIP, ORIENTATION and DEVICE nor GRIP and DEVICE (all p>.1).

Dragging

Fitts law regression – Regression analysis on the experimental data yielded an $r^2 > .99$, an intercept a of .13 s and a slope b of .20 $s.bits^{-1}$. To compare against the overall TouchDB data, we again performed a regression analysis for all the users in the database ($r^2 > .87$, a=.17 s and b=.18 $s.bits^{-1}$) and all the users but without the experimental data ($r^2 > .92$, a=.12 s and b=.17 $s.bits^{-1}$).

Main effects on TIME– As expected, we found a significant main effect of ID ($F_{2,34}$=528.6, p<.0001). Post-hoc tests showed significant differences between all levels of ID (all p<.005) with higher-ID tasks taking longer. We found a significant main effect of DEVICE ($F_{1,17}$=4.6, p<.05). Post-hoc tests showed that participants were significantly faster on the 5-inch device than on the 7-inch device (p<.01). We found no significant main effect of ORIENTATION ($F_{1,17}$=.5, p=.5) or GRIP ($F_{2,34}$=3.1, p=.06).

Interactions on TIME– We found significant interactions between ID and DEVICE, and between ID and GRIP (DEVICE: $F_{2,34}$=5.4, p<.01; GRIP: $F_{4,68}$=4.3, p<.005). Post-hoc tests showed multiple significant differences, but not between a specific ID across the other factors (all p>.3) except between ID 4 across DEVICE (p<.0001), where the 5-inch device was faster than the 7-inch device. We also found a significant interaction between GRIP and DEVICE ($F_{2,34}$=6.2, p<.01). Post-hoc tests showed a difference between $\text{GRIP}^{\text{HOLD\&}}_{\text{TOUCH}}$

Tasks	ID	DEVICE	ORIENTATION	GRIP
Pointing	$p < .0001$	$p = .13$	$p = .5$	$p < .005$
	$F_{3,51} = 216.7$			$F_{2,34} = 8.1$
Dragging	$p < .0001$	$p < .05$	$p = .5$	$p = .06$
	$F_{2,34} = 528.6$	$F_{1,17} = 4.6$		
Rotation	$p < .0001$	$p < .01$	$p < .05$	$p < .0005$
	$F_{3,51} = 127.6$	$F_{1,17} = 9.2$	$F_{1,17} = 7.1$	$F_{2,34} = 12.2$
Scaling	$p < .0001$	$p < .005$	$p = .4$	$p < .005$
	$F_{2,34} = 170.2$	$F_{1,17} = 10.6$		$F_{2,34} = 6.3$

Table 2: Summary of the main effects on TIME of the ANOVA analysis. Tapping is not reported as there were too few levels of ID in common across the combination of other factors.

across DEVICE ($p<.05$), with $\text{GRIP}^{\text{HOLD\&}}_{\text{TOUCH}}$ being faster on the 5-inch screen. We found no significant interaction between ID and ORIENTATION, ORIENTATION and GRIP, or GRIP and DEVICE (all $p>.2$).

Summary

Overall, both pointing and dragging are well described by Fitts Law, and in both tasks, time increases with difficulty. These results were expected, but are useful in validating the data gathered in the field experiment. In addition, our results confirm that there can be significant differences across device form factors. We found that pointing is faster when holding and operating a device with both hands, but this was not the case for dragging. Since pointing is a discrete task, two hands can better cover the screen and thus speed up the interaction - but for dragging, the task requires one finger's interaction and so there is no advantage in having two fingers available. We also found differences across device sizes for the dragging task: dragging on a 5-inch screen was faster than on a 7-inch screen, possibly due to the natural constraint of hand size. Overall, we found similar slope coefficients to previously-reported values [9].

THE STORYBOARD EMPIRICAL MODELLING TOOL

We developed StEM to provide a simple and fast means for predicting user performance with touch interfaces. StEM is inspired by systems like CogTool, which take a complicated modelling framework (KLM) and provide an interactive front end that allows broad access to the underlying formalism. As described above, however, CogTool does not include several kinds of touch actions, and models others as composites of existing mouse-based actions.

StEM allows designers to build touch-interaction sequences on top of visual representations of an interface - whether these are rough sketches, wireframes, screenshots of actual prototypes, or even blank screens. Instead of analysing an interaction sequence and extracting touch operators by hand, StEM allows designers to drag and drop touch actions onto the interface pictures (figure 4a), which places a canvas object representing the action onto the interface, and adds the action to the touch-interaction timeline (figure 4a). To specify the specific properties of an action (*eg.* the size and position of a tap area on the device screen), the user manipulates the canvas object on the interface picture (figure 4c).

Because many tasks require interaction with several screens, the designer can link individual screens together into a scenario (figure 1b). A scenario is therefore the unit at which a designer models a high-level task on a particular device - so, for a given scenario, the size of the device has to be set (*ie.* width and height in millimetres).

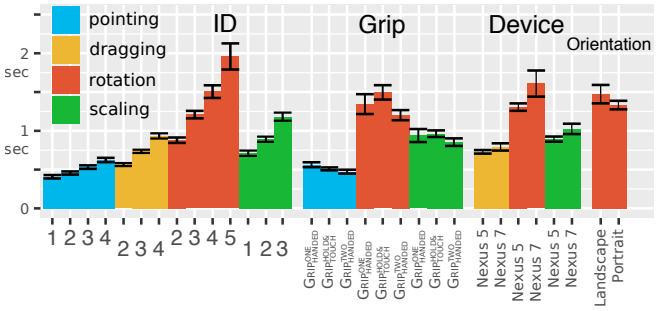


Figure 3: Mean times and 95% confidence interval of each tasks per main factors.

Figure 4: Example of StEM use: (a) the user drags and drops actions onto the timeline to construct the sequence of interaction of the current application screen; (b) the corresponding sequence of operators is automatically updated; (c) the user specifies the parameters of an action by manipulating its canvas representation (*eg.* position, size, ...); (d) after querying the database, the prediction times are displayed as stacked bars (each colour represents an operator).

Once a scenario has been designed, StEM automatically computes the corresponding sequence of operators using a set of predefined rules (described below). Each operator is associated with an index of difficulty, a time, or a movement direction depending on its type. At any time, the designer can query the database and retrieve the predicted time for the current sequence of operators. She can also specify a number of filters (figure 1c) that specialize the predictions to specific devices, screen orientations, or hand grips. In addition, the designer can specify two percentage thresholds to obtain predictions for two user groups: the x% fastest and the y% slowest users in the database.

The prediction results are then displayed in three charts (using the *d3.js* framework). The first (figure 1d) presents the general trend for all users, the fastest users, and the slowest users. The chart displays the total time taken by a sequence as well as the time taken by each operator: the bars stack each operator according to the order of actions in the timeline. The second chart breaks down the first graph to each individual user (bottom left graph on figure 1b). The third graph shows the distribution of the users (bottom right graph on figure 1b).

Operators and rules

The sequence of operators is computed based on previous work [18, 29], except that we use a different definition for tapping compared to Lee *et al.* [18]. Lee considers tapping as a pointing task with a small amplitude (a fixed amplitude of 10 mm was used). In StEM, we consider tapping as a pointing action where the amplitude is unknown (*ie.* where the starting position of the finger is unknown - as described above we use a fixed amplitude based on the screen size). Lee's tapping operator is therefore captured by our pointing operator - for example, pushing a button twice is modeled by a pointing action with an amplitude of 0 mm (*ie.* an ID of 0 bits).

The operators we use in StEM are:

- T (tapping): pressing an on-screen target without knowledge of the starting finger position.
- P (pointing): pressing an on-screen target with knowledge of the starting finger position
- D (dragging): moving an on-screen object until it is within a designated area
- R (rotation): rotating an on-screen object with two fingers
- S (scaling): resizing an on-screen object with a two-finger pinch gesture
- F (flicking): a ballistic linear movement in one of the cardinal directions (up, down, left, right)
- Sw (swiping): a controlled linear movement in one of the cardinal directions
- ● (long-press): a timeout indicating a long press on a touch target (typically 300 ms).
- M (mental) and R (system response): timeout representing the time taken by a user to make a decision and time out representing the time taken by the system to reach its new state. For the sake of simplicity, we gathered M and R into a generic operator W (wait) that can be specified by the designer.

Figure 5 represents the state machine used by StEM to compute the sequence of operators in the action order specified by the user. For instance, describing an icon drag and drop on the second homescreen would result in: the user flicks left to the second homescreen (*Idle* to *Flick* to *Idle*, TF), the user performs a long press on the icon to move (*Idle* to *Draggable object* to *Long-press* to *Draggable object*, T●), the user then moves and drops the icon at the desired location (*Draggable object* to *Drop area*, D).

It is important to note that StEM users do not need to be aware of the underlying models. They just drag and drop widgets on the background (as with tools like Balsamiq©). The interaction sequence and IDs are automatically determined, based on parameters set by StEM users, such as the pointing distance and target size. Users also determine factors influencing the pool of data within the TADB that is used to calculate values. These factors include demographic characteristics (*eg.* age or gender) and usage characteristics (*eg.* grip or orientation). These parameters are specified through filters (left of figure 1c). All the tasks characteristics and filters are taken into account when automatically querying the TADB. Similar to KLM, prediction times are computed by summing time estimates for the interaction atoms. Except for swipe tasks, no average or median times are used for predictions. Intercept and slope values for Fitts' Law calculations (a and b parameters) are dynamically determined based on regression equations that draw from the appropriate pool of data within the TADB. The aggregation method for time used in the Fitts' Law calculations depends on the selected checkbox (mean or median) in the StEM UI.

Case study

As a demonstration of how StEM can enable comparison of design alternatives without full implementations, we present a hypothetical case study in which a designer wants to create

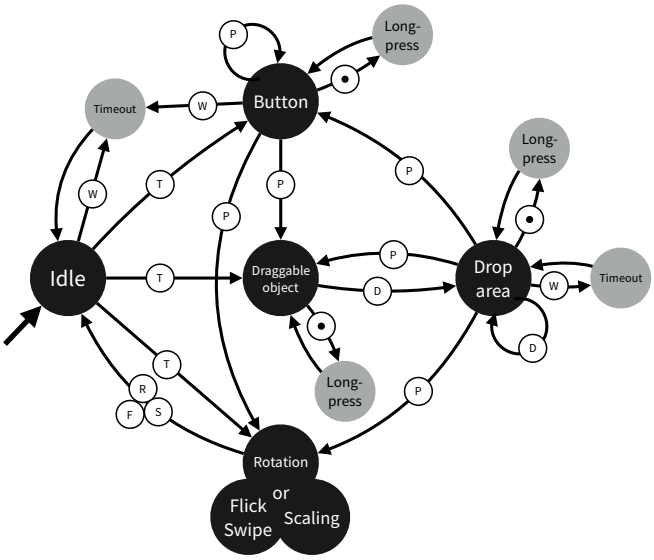
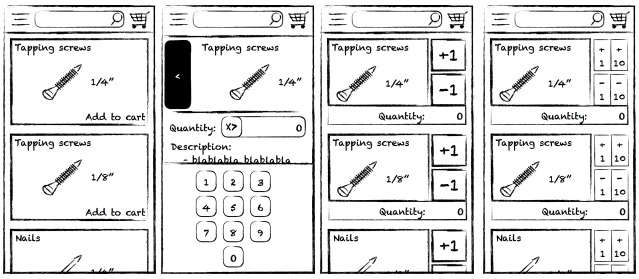


Figure 5: State machine representing the rules used to compute the interaction sequences. Black and grey circles represent the components that can be added to the interaction timeline (*Idle* represents the neutral state). White circles represent the operators of the interaction sequences: T for tapping, P for pointing, D for dragging, R for rotation, S for scaling, Sw/F for swiping/flicking, W for waiting, and ● for a long press. Since rotation, scaling, and swiping/flicking follow the same rules, they were grouped together on the state machine

a shopping application with efficient touch interaction. After looking at existing layouts she decides to compare three designs for adding items to the shopping cart. In design A (figure 6a), the user is shown a list of cards representing items. To buy an item, the user taps the card, which pops up an input box and a numerical keypad that can be used to specify a quantity. In design B (figure 6b), the user is shown the same list of cards, but to enter a quantity she uses a "+1" button located beside the item. In design C (figure 6b), the interface provides both "+1" and "+10" buttons.

Using StEM and a set of wireframe mockups of the designs, the designer predicts the interaction time for buying different quantities (1, 15, 95 and 100) of an item. The prediction results are summarized in table 3. When choosing only one item, design B is the quickest, because the single action can be performed on the first screen and the button is bigger than in design C. When buying 15 items, however, design B be-



(a) Design A **(b) Design B** **(c) Design C**

Figure 6: Designs of the case study.

gins to show its limitations. Design C is still more efficient than design A for 15 items, because bringing up the second screen takes more time than six button-presses. When buying 95 items, design A is the fastest, but is not substantially better than Design C - and when buying 100 items, Design C is once again the quickest choice. StEM's ability to provide quick performance predictions for the different design alternatives can help reveal that adding a second entry screen may not be a better choice, unless customers typically purchase more than 100 items at a time - and in addition, that the larger button on Design B is only marginally better for single purchases, and is dramatically slower for multiple purchases. Depending on the usage context (*ie.* the typical number of items chosen at once), the designer now has a better overview of the alternatives, at only the cost of developing wireframe mockups.

STEM EVALUATION

We assessed the accuracy of StEM's performance predictions in three ways: first, we compared predicted times for specific task scenarios to empirically-recorded values; second, we compared StEM's predictions to two existing tools (CogTool and FLM); and third, we modeled two interaction techniques, Marking Menus and FastTap Menus, and replicated the experimental results of their comparison. In this evaluation we do not test the tool's usability.

Prediction accuracy for task scenarios

In order to evaluate StEM's absolute time predictions, we modelled a diversity of realistic tasks in term of context (home screen, settings, applications), motor actions (pointing, typing, dragging gestures) and task duration (short, long) from real world applications, and then compared StEM's predictions to empirical values. The scenarios were: reorganizing app icons by moving them between Android's fourth and second home screens; checking for system updates in the Android settings menus; answering, deleting and archiving email messages in Gmail; manipulating a map view in Google Maps; and carrying out a shopping task on the Amazon website in the Chrome browser. All scenarios (and our predictions) were based on the interface of a Nexus 5 smartphone.

We asked six participants (mean age 23.8, SD 1.3, 2 females, all daily users of touchscreens) to perform these five scenarios three times each, using a normal pace of interaction. All trials were video-recorded at 60 fps, and task completion times were manually extracted from the video (from the first arm movement to the target application state). We then compared the empirical completion times with those predicted by StEM.

Results

	1 item	15 items	95 items	100 items
Design A	3.9s	4.4s	4.4s	4.7s
Design B	.7s	4.6s	26.7s	28.1s
Design C	.8s	2.3s	4.5s	3.3s

Table 3: Summary of the prediction times for design A, B and C.

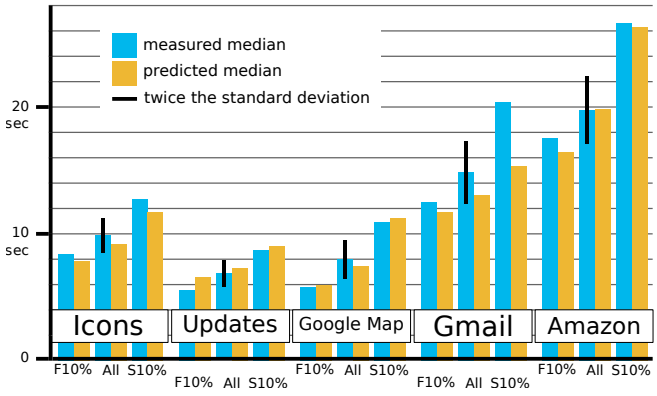


Figure 7: Measured and predicted times for each scenario. The corrections account for the screens swapping. F10% corresponds to the fastest 10% and S10% for the slowest 10%.

In Table 4, we summarize the measured times of interaction as well as StEM's predictions. For each scenario, we report the median of all the 18 trials, the two fastest trials (11% of the total number), the two slowest trials, as well as the standard deviation of all trials. We also report the median times predicted using StEM for all the users in the database, the fastest 10%, and the slowest 10%. We then report the difference between the measured and predicted time, and highlight in bold differences that are more than one standard deviation from the measured time.

To account for the system response time, we measured the time needed by the tablet to open application and swap screen. In Gmail, opening the email thread we used for the experiment took 1.7s, and screen swapping (even though the application was already open) took on average 500ms. We therefore add an W operator to our scenario modelization as follows: we added fixed times for three screen transitions for the Update scenario (W=1.5s), one transition for Google Maps (W=.5s), four for Gmail (W=2s), and two for Amazon (W=1s). We also added a W=1.7s app-loading time to the Gmail scenario. The empirical tasks took between 10 and 20 seconds. In almost all cases, StEM's predictions were off by less than a second and a half. The times are presented in figure 7.

Our predictions (for all users, using the median) differed from empirical values by: 7% for the Icons scenario (0.7s less than the median), 7% for the Updates scenario (0.5s more), 6% for the Google Maps scenario (0.5s less), 13% for the Gmail scenario (1.9s less), and <1% for the Amazon scenario (<0.1s more).

The difference between predicted and empirical values for the slowest users in Gmail was still more than a standard deviation (5.2, SD 2.5s). When looking at the empirical data for the Gmail scenario, we found that three participants were substantially slower during their first trials (user 1 went from 20s to 17.1s and 14.7, user 3 from 20.8 to 15.5 and 15.8, and user 6 from 18.8 to 15.4 and 13.1). Looking at the videos, it appeared that those users were searching for the next action to perform, and were thus still learning about the task. If

	Experiment				StEM			$t_{exp} - t_{StEM}$		
	SD	Fastest	Median	Slowest	Fastest	Median	Slowest	Fastest	Median	Slowest
Icons	1.4	8.4	9.9	12.8	7.8	9.2	11.7	0.6	0.7	1.1
Updates	1.1	5.6	6.8	8.7	6.6	7.3	9.0	-1.0	-0.5	-0.3
Google Maps	1.5	5.8	8.0	10.9	5.9	7.4	11.2	-0.1	0.5	-0.3
Gmail	2.5	12.5	14.9	20.4	11.7	13.0	15.3	0.8	1.9	**5.2**
Amazon	2.7	17.5	19.8	26.6	16.4	19.8	26.3	1.1	0.0	0.3

Table 4: Summary of the interaction times in seconds of each scenario (using the median). To compute the fastest and the slowest times. We took a 10% threshold (*ie.* 2 trials in the experiment). Bold text indicates that predicted times are more than one standard deviation from the measured times.

we consider only the second and third iterations of the task (where participants were familiar, and were therefore primarily carrying out touch actions rather than mental operations), our predictions are closer (from a difference of 3.9s to .2s for all users, and from 5.2s to 1.5s for the slowest participants). In the discussion below, we also consider the limitation that StEM currently only predicts motor actions.

Comparison to predictions from CogTool and FLM

Our second evaluation of StEM was to compare our prediction results with the two best current modelling environments: CogTool and the Fingerstroke-Level Model. We used the available operators and time estimates from CogTool and FLM to model the real-world scenarios described above, using the regression equations provided in [18] and the standard CogTool distribution from `http://github.com/cogtool`. However, for both FLM and CogTool, the Google Maps scenario is not reported because neither tool includes rotation and scaling operators; and for CogTool, swipe actions in the Update, Gmail, and Amazon scenarios are modeled with drags (since CogTool does not include a native swipe operator). Finally, we report only median values for all participants, since neither CogTool nor FLM predict a range of user performance. A summary of results are shown below and in table 5: in all cases but one, predictions from both CogTool and FLM are worse than those of StEM (and in many cases by substantial amounts).

Using FLM models:

- Icons: 5.8s (4.1s less than the median time, off by 41%)
- Update: 5.5s (1.3s less, off by 19%)
- Gmail: 11.0s (3.8s less, off by 26%)
- Amazon: 15.3s (3.48s less, off by 18%)

Using CogTool:

- Icons: 5.5s (4.4s less than the median time, off by 44%)
- Update: 6.7s (.1s less, off by 2%)
- Gmail: 11.4s (3.5s less, off by 23%)
- Amazon: 17.7s (2.0s less, off by 10%)

Replicating a published comparison of touch techniques

For our third evaluation, we re-created an interaction sequence from prior research, which compared the FastTap interaction technique [12] to Marking Menus [16]. We modelled both the novice and expert interaction modes for these techniques, and compared the completion times reported in the published study to those predicted by StEM. In table 6, we summarize the experimental times and standard deviations reported in [12] as well as the predicted mean and median times using StEM. As above, we report the differences between the experimental and predicted times and highlight in bold the differences which are more than one standard deviation apart.

On average, our predictions are off by a third of a second; in only one case (the median time prediction of FastTap used in expert mode) is the StEM prediction off by more than one standard deviation. We hypothesize that this divergence is due to the chorded interaction of FastTap which is not yet fully modelled in StEM. However, despite this limitation, our prediction is able to correctly match the reported order between the techniques, using both mean and median times. In both the empirical study and the StEM predictions, the interfaces performed in the same order: 1) FastTap expert, 2) FastTap novice, 3) Marking Menus expert, and 4) Marking Menus novice. This result provides additional evidence that StEM is able to successfully predict comparisons between design alternatives, even when these alternatives involve novel interaction techniques.

DISCUSSION

StEM is designed to integrate support for rapid prototyping of touch interactions with the benefits of accurate performance prediction. Through drag-and-drop manipulation of touch interaction UI components, StEM users can quickly and easily specify the interaction. Having done so, performance predictions are automatically available, covering a variety of postures and form-factors, as well as enabling the designer to selectively review different performance quantiles, such as the fastest or slowest users.

A large part of the flexibility and range of the StEM tool is based on the development and validation of the data corpus (TADB) that underlies StEM's predictions. While previous

	t_{exp}	StEM	FLM	CogTool
Icons	9.9s	7% (-0.7s)	41% (-4.1s)	44% (-4.4s)
Updates	6.8s	7% (+0.5s)	19% (-1.3s)	2% (-0.1s)
Google Maps	8.0s	6% (-0.5s)	—	—
Gmail	14.9s	13% (-1.9s)	26% (-3.8s)	23% (-3.5s)
Amazon	19.7s	<1% (<0.1s)	18% (-3.5s)	10% (-2.0s)

Table 5: Summary of the prediction errors (with corrections) for StEM, FLM and CogTool methods. Errors are the absolute differences between predicted and empirical times divided by the empirical time.

modelling tools, such as CogTool, have merged prototyping and prediction capabilities, they have lacked the ability to describe many of the components of touch interaction. Although we hope that designers will choose to use StEM (once it is released), the full data corpus is available to researchers and practitioners, enabling them to re-purpose the extensive dataset for their own uses.

Limitations and further work

The TADB data corpus was populated by volunteer participants who carried out cued tasks in their own time on their own devices, and without the controls that can be obtained in laboratory settings. This method has associated strengths and weaknesses. Key strengths are that the corpus can be populated with much a larger and broader dataset than can be pragmatically obtained in the lab, and that the participants' performance may better reflect performance during real interaction (rather than optimal lab conditions). Key weaknesses, however, concern the introduction of noisy and erroneous samples into the dataset. For example, when electing to contribute data to the corpus, participants manually selected a control identifying the posture they would use during upcoming trials, their age, gender, and handedness; if these responses are incorrect, the data for subsequent trials could misrepresent the actual performance in the intended condition. At present we have taken no action to identify or remove noisy samples from the corpus, but doing so may improve modelling accuracy. In general, as crowd-sourcing of experimental data becomes increasingly common there is a pressing need for improved methods for data cleaning [5, 14].

The data corpus also examined only the most frequently occurring atoms of touchscreen interaction, leaving opportunities for extending the range of elemental interactions featuring in the dataset. For example, we have not as yet examined double-tap actions, force-based actions (as used on recent Apple devices), and certain finger-chorded actions (e.g., two-finger tap/swipe or five-finger pinch/spread). All of these more exotic atoms of touch interaction are used in contemporary devices, and there is value in characterizing their performance within future versions of the corpus. A possibility to better capture the diversity of touch interaction would be to investigate programming by demonstration (like in Marquise [26] or FrameWire [20]): demonstrating user actions while using StEM could help to inform researchers about which atoms to integrate in the TADB in order to improve StEM prediction.

		Paper		StEM		$t_{exp} - t_{StEM}$	
		Time	SD	Mean	Median	Mean	Median
Expert	MM	2.2	.8	2.3	2.1	-0.2	0.1
	FT	1.6	.3	1.3	1.2	0.2	**0.3**
Novice	MM	2.9	.9	3.1	3.0	-0.2	-0.1
	FT	2.1	.4	1.9	1.7	0.2	0.3

Table 6: Summary of the comparison between the Marking Menus and the FastTap technique in seconds (using mean and median). In bold are the differences between the reported and predicted times which are greater than the reported standard deviation.

Our predictions only focus on motor action time. Accounting for cognitive processes, such as mental preparation time, time spent searching for a target in a list, or application response time, is left to the designers (who can model these elements with StEM's generic Wait operator). However, modelling such processes in more detail – for example by helping designers choosing the right amount of time for these operators or providing operators accounting for user expertise (the current version of StEM only predicts time for users familiar with the mockup UI) – is worth exploring as future work.

Another useful direction for further work on the corpus is the performance impact of chaining a series of interaction atoms into a larger interactive series. Currently, the TADB corpus provides timing estimates for each discrete interface action that it models, and StEM produces predictions based on the sum of these timing estimates. By modifying our method for gathering corpus data we could validate whether combined operations are best modelled through simple timing summation, or whether additional parameters are needed to account for any difference between observed performance and the simple summation model.

Finally, a key area for further work concerns evaluating the effectiveness of StEM as a design tool. While the corpus has independent research value, our overall research objective is to aid design by linking eased prototyping with largely automatic performance prediction. The work presented in this paper suggests that the performance predictions are relatively accurate, and StEM has several features that should contribute to a user experience that is similar to existing design tools, but further studies are needed to examine how designers use StEM and whether they find it useful and usable for improving their designs. We intend to begin this evaluation by examining novice designers in undergraduate classes, and then shift our focus to professional designers. More generally, we would like to try using StEM as a teaching tool and evaluate if it conveys insights on layout and interaction designs.

CONCLUSION

Storyboard Empirical Modelling (StEM) is a drag-and-drop tool that allows designers to quickly prototype touch interactions and explore their performance implications. StEM relies on the Touch-Action database, a crowd-sourced data corpus that provides empirical characterisations of tap, point, drag, swipe, scale and rotate touch interactions across a wide range of device types, hand grips, and screen orientations. Although we developed the TADB corpus primarily as the foundation for predictions with StEM, the corpus is available on the web for other researchers and practitioners. We carried out several evaluations of StEM, and showed that its predictions are accurate within an average of 7% difference from empirical values, and never worse than 13%, across a variety of scenarios – and substantially better than existing tools. StEM provides new capabilities for designers and researchers who need to understand user performance with touch interfaces at any stage in the design process.

REFERENCES

1. Evgeniy Abdulin. 2011. Using the Keystroke-level Model for Designing User Interface on Middle-sized Touch Screens. In *Proc. of CHI EA*. ACM, 673–686. http://doi.acm.org/10.1145/1979742.1979667
2. Caroline Appert, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2005. Context matters: Evaluating Interaction Techniques with the CIS Model. In *People and Computers XVIII — Design for Life: Proceedings of HCI 2004*. Springer London, 279–295. http://dx.doi.org/10.1007/1-84628-062-1_18
3. Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts Law: Modeling Finger Touch with Fitts' Law. In *Proc. of CHI*. ACM, New York, NY, USA, 1363–1372. http://dl.acm.org/citation.cfm?id=2470654.2466180&coll=DL&dl=ACM&CFID=244634215&CFTOKEN=16005854
4. Stuart K Card, Thomas P Moran, and Allen Newell. 1980. The keystroke-level model for user performance time with interactive systems. In *Communications of the ACM*. ACM, 396–410. http://doi.acm.org/10.1145/358886.358895
5. Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing. In *Proc. of SIGMOD*. ACM, 1247–1261. http://doi.acm.org/10.1145/2723372.2749431
6. A. Cockburn, D. Ahlström, and C. Gutwin. 2012. Understanding Performance in Touch Selections: Tap, Drag and Radial Pointing Drag with Finger, Stylus and Mouse. *Int. J. Hum.-Comput. Stud.* 70, 3 (March 2012), 218–233. DOI: http://dx.doi.org/10.1016/j.ijhcs.2011.11.002
7. Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47, 6 (1954), 381.
8. Clifton Forlines, Daniel Wigdor, Chia Shen, and Ravin Balakrishnan. 2007. Direct-touch vs. Mouse Input for Tabletop Displays. In *Proc. CHI*. ACM, 647–656. http://doi.acm.org/10.1145/1240624.1240726
9. Alix Goguey, Mathieu Nancel, Géry Casiez, and Daniel Vogel. 2016. The Performance and Preference of Different Fingers and Chords for Pointing, Dragging, and Object Transformation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4250–4261. DOI: http://dx.doi.org/10.1145/2858036.2858194
10. Alix Goguey, Daniel Vogel, Fanny Chevalier, Thomas Pietrzak, Nicolas Roussel, and Géry Casiez. 2017. Leveraging finger identification to integrate multi-touch command selection and parameter manipulation. In *IJHCS journal*, Vol. 99. Elsevier, 21 – 36. http://dx.doi.org/10.1016/j.ijhcs.2016.11.002
11. Alix Goguey, Julie Wagner, and Géry Casiez. 2015. Quantifying Object- and Command-Oriented Interaction. In *Human-Computer Interaction – INTERACT 2015: 15th IFIP TC 13 International Conference, Bamberg, Germany, September 14-18, 2015, Proceedings, Part IV (INTERACT '15)*. Springer International Publishing, Cham, 231–239. DOI: http://dx.doi.org/10.1007/978-3-319-22723-8_18
12. Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. 2014. Faster Command Selection on Tablets with FastTap. In *Proc. of CHI*. ACM, 2617–2626. http://doi.acm.org/10.1145/2556288.2557136
13. Paul Holleis, Friederike Otto, Heinrich Hussmann, and Albrecht Schmidt. 2007. Keystroke-level Model for Advanced Mobile Phone Interaction. In *Proc. of CHI*. ACM, 1505–1514. http://doi.acm.org/10.1145/1240624.1240851
14. H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. 2014. Big Data and Its Technical Challenges. (2014), 86–94. http://doi.acm.org/10.1145/2611567
15. Bonnie E John. 2010. Reducing the variability between novice modelers: Results of a tool for human performance modeling produced through human-centered design. In *Proc. of BRIMS*. Springer, 22–25.
16. Gordon Kurtenbach and William Buxton. 1991. Issues in Combining Marking and Direct Manipulation Techniques. In *Proc. of UIST*. ACM, 137–144. http://doi.acm.org/10.1145/120782.120797
17. James A. Landay and Brad A. Myers. 1995. Interactive Sketching for the Early Stages of User Interface Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 43–50. DOI: http://dx.doi.org/10.1145/223904.223910
18. Ahreum Lee, Kiburm Song, Hokyoung Blake Ryu, Jieun Kim, and Gyuhyun Kwon. 2015. Fingerstroke time estimates for touchscreen-based mobile gaming interaction. In *Human Movement Science Journal*. Elsevier, 211–224. http://dx.doi.org/10.1016/j.humov.2015.09.003
19. Hui Li, Ying Liu, Jun Liu, Xia Wang, Yujiang Li, and Pei-Luen Patrick Rau. 2010b. Extended KLM for Mobile Phone Interaction: A User Study Result. In *Proc. of CHI EA*. ACM, 3517–3522. http://doi.acm.org/10.1145/1753846.1754011
20. Yang Li, Xiang Cao, Katherine Everitt, Morgan Dixon, and James A. Landay. 2010a. FrameWire: A Tool for Automatically Extracting Interaction Logic from Paper Prototyping Tests. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 503–512. DOI: http://dx.doi.org/10.1145/1753326.1753401

21. Wendy E. Mackay. 2002. Which Interaction Technique Works when?: Floating Palettes, Marking Menus and Toolglasses Support Different Task Strategies. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '02)*. ACM, New York, NY, USA, 203–208. DOI: http://dx.doi.org/10.1145/1556262.1556294

22. I Scott MacKenzie. 1989. A note on the information-theoretic basis for Fitts' law. *Journal of motor behavior* 21, 3 (1989), 323–330.

23. I. Scott MacKenzie. 1992. Fitts' Law As a Research and Design Tool in Human-computer Interaction. *Hum.-Comput. Interact.* 7, 1 (March 1992), 91–139. DOI: http://dx.doi.org/10.1207/s15327051hci0701_3

24. I Scott MacKenzie. 2013. A Note on the Validity of the Shannon Formulation for Fitts' Index of Difficulty. *Open Journal of Applied Sciences* 3, 06 (2013), 360.

25. Mark Micire, Martin Schedlbauer, and Holly Yanco. 2007. Horizontal selection: An evaluation of a digital tabletop input device. *In Proc. AMCIS* (2007), 164. http://robotics.cs.uml.edu/fileadmin/content/publications/2007/MJM.AMCIS.2007.MTU.final.pdf

26. Brad A. Myers, Richard G. McDaniel, and David S. Kosbie. 1993. Marquise: Creating Complete User Interfaces by Demonstration. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 293–300. DOI: http://dx.doi.org/10.1145/169059.169225

27. Nihan Ocak and Kursat Cagiltay. 2016. Comparison of Cognitive Modeling and User Performance Analysis for Touch Screen Mobile Interface Design. In *International Journal of Human–Computer Interaction*. Taylor and Francis, 1–9. http://dx.doi.org/10.1080/10447318.2016.1274160

28. Manuela Quaresma. 2012. Assessment of visual demand of typical data entry tasks in automotive navigation systems for iPhone. In *Work journal*, Vol. 41. IOS Press, 6139–6144. http://doi.org/10.3233/WOR-2012-1074-6139

29. Andrew D. Rice and Jonathan W. Lartigue. 2014. Touch-level Model (TLM): Evolving KLM-GOMS for Touchscreen and Mobile Devices. In *Proc. of ACM SE*. ACM, Article 53, 6 pages. DOI: http://dx.doi.org/10.1145/2638404.2638532

30. Andrew Sears and Ben Shneiderman. 1991. High Precision Touchscreens: Design Strategies and Comparisons with a Mouse. *Int. J. Man-Mach. Stud.* 34, 4 (April 1991), 593–613. DOI: http://dx.doi.org/10.1016/0020-7373(91)90037-8

31. Jian Zhao, R. William Soukoreff, and Ravin Balakrishnan. 2015. Exploring and modeling unimanual object manipulation on multi-touch displays. *International Journal of Human-Computer Studies* 78, 0 (2015), 68 – 80. DOI: http://dx.doi.org/10.1016/j.ijhcs.2015.02.011